2005

Cognitive Complexity Factors in Data Modeling

Dinesh Batra
Florida International University, batra@fiu.edu

Cognitive Complexity Factors in Data Modeling

Dinesh Batra
Florida International University
Miami, FL 33199
batra@fiu.edu

ABSTRACT

We live in the age of complexity. Yet, complexity is rarely studied in MIS. The paper studies cognitive complexity in an important IS domain – data modeling. The focus is on novice data modelers. Four major sources of complexity principles are identified: *problem solving principles, design principles, information overload*, and *systems theory*. Based on prior literature, the factors that lead to complexity are listed in each category. Each factor is then applied to the context of data modeling to gauge the extent to which it affects data modeling complexity. Redundant factors from different sources are ignored, and closely linked factors are identified. The factors are then integrated to come up with a comprehensive list of factors, which are divided into two categories – those that are intrinsic to data modeling and are difficult to control, and those than can be addressed to minimize data modeling complexity.

Keywords

Data modeling, complexity, problem solving, design principles, information overload, systems theory.

INTRODUCTION

According to Reeves (1996), the correct name for the current age should be the Age of Complexity. Yet, complexity is rarely studied in MIS. In this paper, complexity is examined in a narrow domain – data modeling. The paper attempts to answer the question – what causes complexity in conceptual and logical data modeling? The scope of this study is limited to novice designers given that expert designers, by definition, are able to tackle difficult problems.

The motivation of the study is derived from several studies (e.g., Bock and Ryan, 1993) that found low performance of designers in modeling certain facets such as unary and ternary relationships. Batra and Antony (1994) examined designer performance in modeling open-ended exercises and attributed the low designer performance not just to certain selected data modeling facets but all facets including binary relationships. They found that when designers are asked to solve problems, they commit errors because of biases resulting from naïve heuristics. Size of the problem seemed a major source of complexity, but so were other factors like the way the problems were worded. Batra and Wishart (2004) found that the size of the problem, and all kinds of relationships but specifically higher degree relationships led to lower designer performance. None of the studies, however, has taken a formal or detailed look at what causes data modeling complexity.

To study cognitive complexity in data modeling, we need to first study theoretical or formal sources that have addressed complexity in general. Two references (Reeves, 1996; Reeves, 1999) provide generic and a relatively complete list of factors relevant to the issue. Based on these texts, four sources of complexity principles are identified: *problem solving, design complexity, information overload*, and *systems theory*. The two texts list the specific causes of complexity within each of the sources.

PROBLEM SOLVING PRINCIPLES

This source is based on the work on heuristics (Polya, 1985) and human problem solving (Newell and Simon, 1972). Problem solving is defined as a process of search through the decision space looking for the right operator to transform the problem space into a solution. Cognitively complex problems are those that require a more difficult search through a more complicated maze of possible operators. Complexity then is a matter of the difficulty in finding the right operators that will eventually lead to the ultimate solution.

There are four general ways of problem solving (Anderson, 1985; Krippner and Dillard, 1988; Newell and Simon, 1972; Reeves, 1996):

1. Following algorithms and rules, or specific steps that are known to lead to the solution or goal representation under specific circumstances

2. Following heuristics, or general steps that help to activate memory and have a high probability of generating a solution
3. Following creative techniques that help a person represent the problem in a way that stimulates existing operators in new ways
4. Increasing the availability of relevant declarative knowledge.

In evaluating complexity factors related to problem solving, these general ways of problem solving need to be considered. Based on Funke's (1991) list of elements, the following elements lead to complexity: intransparency, multiple goals, high interactions, connectivity, dynamic nature of problems, and time delay. Out of these, intransparency, which causes complexity because only some variables lend themselves to direct observation, is not relevant to the context.

a) *Multiple goals*: With multiple goals, some may be contradictory, and trade-offs are often required. This is a concern in physical data modeling as the goal of controlling redundancy is matched off against efficiency. However, this is not a source of cognitive complexity in conceptual and logical data modeling.

b) *Domain complexity*: This may conflict with the limited capacity of the problem solver to think it through. If the analyst/designer does not understand the requirements accurately because the domain is complex and requires lot of knowledge, data modeling would obviously suffer. However, user-analyst interaction should enable an analyst to understand the domain, and this factor is not considered as a significant source of complexity.

c) *Connectivity*: Complex problems often contain a high degree of connectivity or interrelationships. Note that connectivity here does not mean cardinality, but refers to the large number of relationships among a limited number of entities. Consider, for example, a requirement from a vehicle sale application: A customer purchases a vehicle with options from a salesperson. The requirements are likely to lead to the following entities: Customer, Purchase (or Sale), Vehicle, Option, and Salesperson. The problem is that all entities are interrelated. Even if we consider binary links only, there are 10 combinations even before cardinalities are considered (Figure 1). This is also the novice designer view and has empirical support given the way novices arbitrarily choose relationships (Batra and Antony, 1994). However, this is not the correct data modeling solution. Based on the semantics of the application, the designer needs to select certain relationships out of all possible relationships. Supplementary information provided with the application requirements (e.g., each purchase involves only one customer) can be used to arrive at the correct solution. But since novice designers use naïve heuristics, their solutions are arbitrary and unpredictable even though the semantics usually dictate a single correct solution.

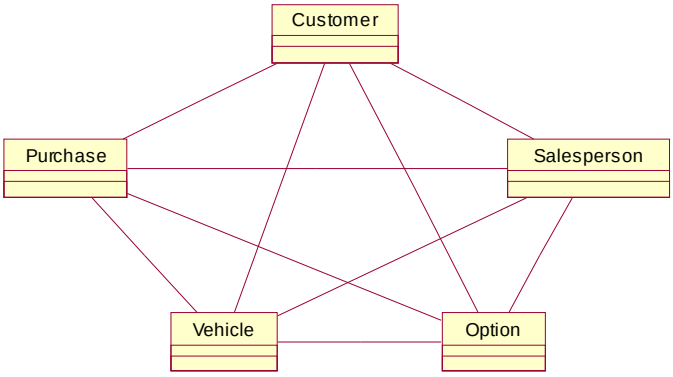

Figure 1: The novice designer view of relationships among five interrelated entities

So how does an experienced designer choose the correct associations among the large number of relationships possible? The designer must come up with a normalized and minimal solution without losing any semantics in the process. There is little empirical work done in this area, so one can only speculate based on general problem solving literature. The experienced designer may follow any of the four problem solving approaches mentioned earlier. For example, the designer many use the notion of patterns after noting that Purchase is a transaction, and that the transaction event has a predictable structure. Further, she may have encountered a similar situation before, and can use analogy to create an initial solution. Experienced designers have a better understanding of data modeling principles and consequences of a bad design. They can use the supplementary information provided with the application requirements to represent relationships that must be modeled, to weed out relationships that are not required, to determine relationships that are derived, and to come up with a correct solution like the one shown in Figure 2.

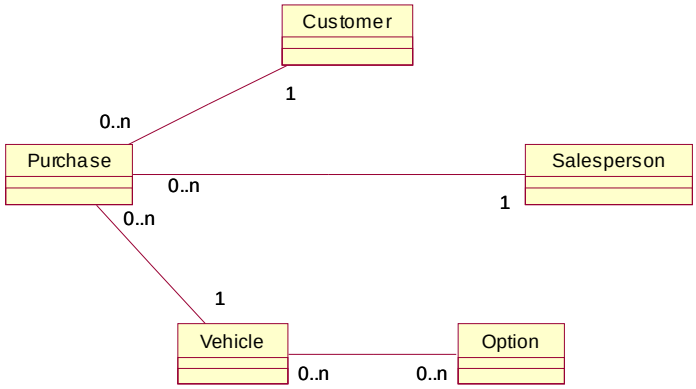

Figure 2: The experienced designer solution

To allow novice designers achieve better designer performance, Batra and Zanakis (1994) proposed an algorithm, mostly rule-based, for data modeling. The algorithm is based on Armstrong's axioms (Maier, 1988) but uses entity relationship (ER) model to convey the approach in simple terms. It addresses the problem caused by high interactions among entities.

d) *Dynamic nature of problems*: The dynamic nature of problems is a factor that is extrinsic and is unlikely to cause complexity in data modeling. If user requirements are constantly changing, it may cause additional work for the designer, but will probably not directly add to the complexity of data modeling. However, dynamic problems are sometimes inherently found in complex domains, and it is the difficulty of understanding the domain that may result in complexity. The dynamic nature of problems factor is ignored in this study.

e) *Time delay*: In data modeling problems, there is a delay between the action taken and the response or the appearance of consequences. This is a not a problem in query writing, but is certainly a problem in data modeling. In query writing, because of immediate feedback and a fair amount of flexibility in breaking down a query into manageable parts, the cognitive complexity of a problem can be controlled. In data modeling, unless prototyping is done immediately and testing is continuous, there is a large time gap between modeling and implementation. The consequences of a bad design are not immediately apparent. Even when the prototyping is done immediately, it still takes some time and expense. Time delay may not directly cause complexity, but it certainly exacerbates the deleterious effects arising from a bad data model.

In summary, the following problem solving related factors cause data modeling complexity: high interactions, which is closely related to connectivity, and time delay.

DESIGN COMPLEXITY

In *User Centered System Design*, Norman and Draper (1986) have included articles that list important design principles for objects used everyday. There are additional principles suggested by Nielsen and Molich (1990). From Norman's (1988) list of good design principles, we can deduce the following elements of complexity: insufficient information, extensive use of memory, no mental aids, no visual feedback, no visual display of what actions are possible, response incompatibility, lack of constraints, no flexibility for error, and no standards. The factor insufficient information is ignored because it is assumed that the relevant information is available. The factor no visual feedback is generalized to the factor no feedback, and the factor no visual display of what actions are possible is generalized to the more generic no visual display. The factors response incompatibility and no standards are considered irrelevant to the context.

f) *Extensive use of memory*: According to Miller (1956), memory overload can occur anytime the number of items to be tracked exceeds the magic number seven. The main cause of memory overload is connectivity, which has been discussed under problem solving factors in the previous section. Since a limited number of entities can result in large number of possible relationships, far greater than the magic number seven, it requires effort-managing strategies to reduce the load on memory. Memory overload can also be caused in determining cardinality especially of higher degree relationships. Note that this factor is related to the factor significant interactions mentioned under problem solving.

g) No mental aids: This factor is prominent in the relational model, which provides practically no mental aid to help in data modeling. Relational databases follow the normalization approach, which prescribes certain properties that databases need to possess. In other words, normalization provides criteria that can be used to assess the quality of a database after data modeling has been done. Thus, normalization is more of an end point check rather than a process that can serve as a mental aid. The ER model provides a visual representation, but there are still no mental aids. If memory overload is to be reduced, mental aids can be provided in terms of simple rules that a novice designer can follow easily. Such rules have been incorporated in the software CODASYS (Antony and Batra, 2002) that provides several mental aids to assist novice designers.

h) *No feedback*: Complexity can result if there is no feedback on the results of an action. This point is similar to the "time delay" issue listed in the previous category. Data modeling provides no immediate feedback, and the consequences of a bad design show up only at the end of the implementation. Prototype testing can mitigate but not prevent this problem.

i) No visual display: It has been well established now that graphical models like the entity relationship (ER), as compared to text based models like the relational, lead to better designer performance. This is because graphical models are vastly superior in showing links, and it is well known that relationships are the leading cause of data modeling complexity. Relationships can be easily represented using links.

j) *Unconstrained choices*: Lack of constraints allows the designer choices among too many options. This is a common problem in data modeling as was illustrated when discussing connectivity in the previous section. In data modeling, there is a vast gap between the problem space and solution space. Constraints facilitate pruning of the problem space so that the solution space can be reached. Algorithms should be geared to effectively utilize the constraints to systematically prune the problem space.

k) *No flexibility for errors*: Once the requirements have been articulated unambiguously, there is not much flexibility in data modeling to allow for alternative solutions. The problem is exacerbated because of lack of feedback. When querying, a user can assess the result and gauge if the query needs to be corrected. Also, a query can be decomposed into cognitively manageable steps. As mentioned under "Time delay" and "No feedback", errors in data modeling are not immediately apparent.

In addition to Norman's design principles, there are additional heuristics suggested by Nielsen and Molich (1990) as good design practices: the use of natural dialogue, using terms familiar to the expected audience, minimizing need for memory, being consistent in all aspects, always providing feedback, using constraints to avoid errors, clearly marking exits, and providing shortcuts. Note that these desirable factors are worded opposite to complexity factors. Many of these factors have been covered. The factors on memory, feedback, and unconstrained choices have already been discussed. The terms "the use of natural dialogue" and "using terms familiar to the expected audience" are similar.

l) *Lack of natural dialogue*: This is again prominent in relational data modeling, which relies on the notion of functional dependency and multivalued dependency. This is clearly an unnatural dialogue for the designer. For example, the definition of the fourth normal form in terms of multivalued dependencies is totally alien to designers. Even the entity relationship concepts like cardinality may be difficult for novice designers. Data modeling methods should rely on language that is close to the way they understand the world. Further, a data model under construction can be interpreted in natural language so that the representation can be better understood and errors can be caught.

m) *Lack of clearly marked exits*: When is a data model complete? There is no clearly marked exit. If, say, all seven entities in a data model have been somehow connected, is it complete? Since all entities do not have to be connected for a data model to be complete, and since some entities may be connected among each other more than once, there is no milestone that signals the end of data modeling. This is a difficult problem, and the best heuristic available today is still the observation that all entities have been connected using relationships.

n) *Lack of shortcuts*: Data modeling should employ heuristics that provide shortcuts that considerably reduce the complexity. It is possible that this may sometimes result in erroneous solutions that need to be corrected after working with a prototype. Nevertheless, heuristic based methods that use short cuts will be more readily understood and used by designers. Further, one can use patterns especially in the case of modeling transactions. At times, the template nature of patterns will provide solutions that are erroneous in the first pass, so the pattern-based approaches need to be integrated with other approaches.

INFORMATION OVERLOAD

Information overload is simply the excessive amount of information that an individual may encounter and which may cause stress and anxiety (Wurman, 1989). The following factors are reported in Reeves (1996) to cause information overload: disorder, novelty, inconsistency, noise, and undifferentiated features.

o) *Disorder*: This pertains to lack of categories. Attribute based models like the relational data model can create information overload and lead to low designer performance. But data models like ER model are based on the notion of abstraction and reduce the complexity by a significant factor. Attributes are first categorized into entities, which are then related thus considerably reducing the number of interrelationships to be evaluated by the designer. Modeling approaches that further abstract entities into clusters (Teorey, Wei, Bolton, and Koenig, 1989) can further reduce complexity by providing layers of abstraction that can lead to better understanding and ease modeling of a large application.

p) Novelty: This pertains to situations that are new and unfamiliar to the designer. This factor is similar to the domain complexity factor discussed under problem solving. A new situation may not cause much complexity if an analogue can be found. However, in general, it will cause some complexity because the solution has to be mostly worked from the first principles. Whether this source is overrated in its contribution to complexity is an empirical issue. One may argue that if the requirements are clearly laid out, and the designer is able to grasp the domain, a data-modeling problem may not be seriously affected by the novelty factor.

q) Inconsistency: Models, by definition, are abstract representations. Further, data modeling is governed by rules that control redundancy and ensure that queries on a data model will not lead to spurious results. Such factors can sometimes cause representation inconsistency between the real world semantics and their data model representations. For example, a supervisor-subordinate relationship is between two persons, but the data model representation is not binary; instead it is unary. Further, a data modeling relationship may seem inconsistent with the real world view. For example, in Figure 1, if the options can be changed every time a vehicle is sold, the transaction Purchase needs to connect to the Option entity. This may seem inconsistent with the real world view that suggests options belong to the vehicle. Inconsistency may also result when similar kinds of relationships have very different representations. For example, in the relational model, a many-many relationship requires a separate relation; a one-many relationship does not. However, the factor undifferentiated features discussed later better handles this inconsistency.

r) Noise: This pertains to the presence of irrelevant information. In data modeling, noise can take a totally different meaning since credible information may also be considered as noise. Consider the example that involves a student's registration into sections of courses. In user requirements (e.g., use cases), it would be normal to write that a student registers for a course suggesting there is a business relationship between the two; yet, more details and minimality requirements will reveal that they don't have a direct data modeling relationship, which actually goes from student to registration, registration to section, and section to course (Figure 3).

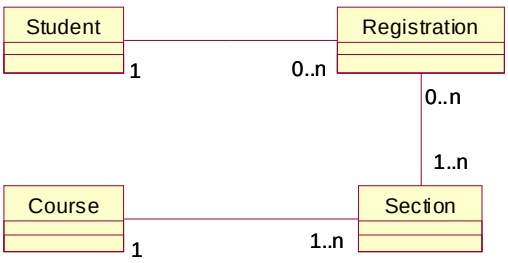

Figure 3: Student Registration example

Note that a use case may mention about student taking courses as much as about student enrolling in section, or registration involving courses. Although these would be well accepted in user requirements, there is no direct relationship capturing these statements. In data modeling terms, these statements constitute noise that can actually mislead the designer into modeling faulty relationships (Batra and Antony, 1994). The indirect relationships are established using queries. However, a novice designer may not be able to make the distinction.

s) *Undifferentiated features*: This pertains to the situation when features are not distinct. Data models like the relational don't have separate representations for different facets; everything is handled using relations. This can make modeling difficult since different semantics need to be squeezed into the same structure. Data models like the extended entity relationship (EER) model (Teorey et al, 1986) provide the rich set of constructs to provide appropriate constructs for different semantics.

SYSTEMS THEORY

Complexity has also been addressed from the systems viewpoint (Banathy, 1991; Flood and Carson, 1990). The following factors, most of which have already been covered or are not directly relevant to the problem, have been reported to increase complexity: significant interactions, high number of parts, nonlinearity, broken symmetry, lack of constraints, open versus closed to their environment, human versus machine, and emergence – characteristics of a whole different from its parts. Out of these, the factors nonlinearity, broken symmetry, open versus closed to their environment, human versus machine, and emergence do not seem relevant. The factor significant interactions and lack of constraints have been considered earlier. Only one remaining factor – high number of parts – seems relevant.

t) *High number of parts*: This factor does increase complexity, but not in the way it does in other contexts like programming. It is agreed that as the number of parts increases, complexity rapidly increases because of the number of interactions. In the Problem Solving section, it was shown that the number of relationships increases rapidly (actually at a combinatorial rate) with respect to an increase in the number of entities. So, how is it possible that a large database, which may easily have more than 100 entities, gets modeled at all? This is possible because a given entity is not related to each and every entity in the application. Typically, an entity closely relates to only a small set of entities. Although even a small set like 7 or 8 entities can give rise to a large number of relationships, at least this puts some limit on the possible number of relationships. Closely related entities form a cluster (Teorey et al, 1989). Thus, a high number of parts may not be such a critical factor in data modeling complexity, which may be more dependent on the number of entities in a cluster. Further, it is unlikely that a novice designer would be asked to model a very large database.

INTEGRATING THE COMPLEXITY FACTORS

The paper has listed factors from four sources of complexity principles - problem solving, design complexity, information overload, and systems theory – and examined the factors in the context of data modeling. Since some factors are same or closely related among these sources while others are unique within a source, the factors need to be compared and integrated. The factors are tabulated (see Table 1) so that a comprehensive list of factors can be obtained. The first four columns list these factors, and the fifth provides the most appropriate name. Across a row or within a cell, the factors are clustered based on similarity. The table is divided into kinds of factors: those that largely *cannot* be controlled, and those that can be partially or fully controlled. The exercise results in a comprehensive list of factors that cause complexity in data modeling. The ones that cannot be controlled are: significant interactions, no feedback, limited solutions, novelty, and semantic mismatch. The ones that can be partially or fully controlled are: no mental aids, no visual display, lack of natural dialogue, lack of clearly marked exits, lack of shortcuts, low abstraction, and undifferentiated features.

Significant interactions is the basic factor that causes cognitive complexity in data modeling. It cannot be controlled because the large number of interactions among a limited number of entities is a given. What can be done is to mitigate the cognitive complexity by providing consulting support to the novice designer. Another important factor is the lack of feedback when conducting data modeling. Prototyping can indirectly mitigate problems associated with the long lead times between modeling and implementation. There is not much that can be done about the very limited number of data modeling solutions for a given application. The factors novelty and inconsistency are less serious.

Problem Solving	**Design**	**Information Overload**	**Systems Theory**	**Proposed Factor**
Factors that largely cannot be controlled				
Connectivity	Unconstrained choices Extensive use of memory	Noise	Significant interactions Lack of constraints High number of parts	Significant interactions
Time delay	No feedback			No feedback
	No flexibility for errors			Limited Solutions
		Novelty		Novelty
		Inconsistency		Semantic mismatch
Factors that can be partially or fully controlled				
	No mental aids			No mental aids
	No visual display			No visual display
	Lack of natural Dialogue			Lack of Natural Dialogue
	Lack of clearly marked exits			Lack of clearly marked exits
	Lack of shortcuts			Lack of shortcuts
		Disorder		Low abstraction
		Undifferentiated features		Undifferentiated features

Table 1: Integrating the complexity factors

Other factors can be partially or fully controlled. Visual display is now routinely provided through the use of ER or UML diagramming. Prototype tools that provide mental aids have been tested in laboratory settings. Similar tools can address the lack of natural dialogue. Although there is no immediate solution to the lack of clearly marked exits issue, simple heuristics can partially address the problem. Shortcut methods have been specified to mitigate the cognitive strain caused by significant interactions. Clustering techniques can provide layers of abstractions. Tools that translate from a conceptual to a logical data model can address the undifferentiated features problem.

CONCLUSION

The paper has considered four viewpoints of cognitive complexity – problem solving principles, design principles, information overload, and systems theory – to list factors that cause complexity in data modeling. A large number of factors have been discussed. Some factors are intrinsic to data modeling and are beyond a researcher's or developer's control. Other factors can be addressed to manage and ease complexity. Evaluating severity rating for each factor can also extend this paper. Further, this information can be used to come up with an instrument for measuring complexity. Future research can also investigate the associations among the factors.

REFERENCES

1. Anderson, J. R. (1985) *Cognitive Psychology and its Implications.* New York: W.H. Freeman.
2. Antony, S. and Batra, D. (2002) CODASYS: A Consulting System for Conceptual Database Design *The Data Base for Advances in Information Systems*, 33, 3, 54-68.
3. Banathy, B.H. (1991) *Systems design of education*. Englewood Cliffs, NJ: Educational Technologies Publications.
4. Batra, D. and Antony, S.R. (1994) Novice errors in database design. *European Journal of Information Systems*, 3, 1, 57-69.
5. Batra, D. and Zanakis, S.H. (1994) A Conceptual Database Design methodology based on Rules and Heuristics. *European Journal of Information Systems*, 3, 3, 228-239.
6. Batra, D., & Wishart, N.A. (2004). Comparing a rule-based approach with a pattern-based approach under varying task complexity in conceptual data modeling. *International Journal of Human Computer Interaction, 61*(4), 397-419.
7. Bock, D. and Ryan, T. (1993) Accuracy in Modeling with Extended Entity Relationship and O-O Data Models, *Journal of Database Management* 4, 4, 30-39.
8. Flood, R. and Carson, E. (1990) *Dealing with complexity*. New York: Plenum.
9. Funke, J. (1991) Solving complex problems: Exploration and control of complex social problems. In R. Sternberg & P. Frensch (Eds.) *Complex Problem Solving* (pp. 185-222). Hillsdale, NJ: Lawrence Erlbaum.
10. Krippner, S. and Dillard, J. (1988) *Dreamworking: How to use your dreams for creative problem-solving*. Buffalo, NY: Bearly Limited.
11. Liao, C.C. and Palvia, P.C. (2000) The impact of data models and task complexity on end-user performance: an experimental investigation, *International Journal of Human-Computer Studies*, 52(5), 831-845.
12. Miller, G. (1956) The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63(1), 81-97.
13. Newell, A. and Simon, H.A. (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall.
14. Nielsen, J. & Molich, R. (1989). Teaching user interface design based on usability engineering. *SIGCHI Bulletin*, 21, 1, 45-48.
15. Norman, D. (1988) *The psychology of everyday things*. New York: Basic Books.
16. Norman, D. and Draper, S. (Eds.) (1986) *User Centered System Design.* Hillsdale, NJ: Lawrence Erlbaum.
17. Polya, G. (1985) *How to solve it*. Princeton, NJ: Princeton University Press.
18. Teorey, T., Wei, G., Bolton, D.L., and Koenig, J.A. (1989) ER model clustering as an aid for user communication and documentation in database design, *Communications of the ACM*, 32, 8, 975-987.
19. Reeves W.W. (1996) *Cognition and Complexity: The Cognitive Science of Managing Complexity*, London: Scarecrow Press.
20. Reeves W.W. (1999) *Learner-Centered Design*, California: Sage Publications.
21. Wurman, R.S. (1989*) Information Anxiety*. New York: Random House.